Using a Workflow Management Platform in Textual Data Management

Triet Ho Anh Doan[†], Sven Bingert & Ramin Yahyapour

Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen, 37077 Göttingen, Germany

Keywords: Flowable; workflow; text analysis; knowledge graph; persistent identifier

Citation: Doan, T.H.A., Bingert, S., Yahyapour, R.: Using a workflow management platform in textual data management. 4(2), 398-408 (2022). doi: 10.1162/dint_a_00139

Received: August 16, 2021; Revised: November 28, 2021; Accepted: February 5, 2022

ABSTRACT

The paper gives a brief introduction about the workflow management platform, Flowable, and how it is used for textual-data management. It is relatively new with its first release on 13 October, 2016. Despite the short time on the market, it seems to be quickly well-noticed with 4.6 thousand stars on GitHub at the moment. The focus of our project is to build a platform for text analysis on a large scale by including many different text resources. Currently, we have successfully connected to four different text resources and obtained more than one million works. Some resources are dynamic, which means that they might add more data or modify their current data. Therefore, it is necessary to keep data, both the metadata and the raw data, from our side up to date with the resources. In addition, to comply with FAIR principles, each work is assigned a persistent identifier (PID) and indexed for searching purposes. In the last step, we perform some standard analyses on the data to enhance our search engine and to generate a knowledge graph. End-users can utilize our platform to search on our data or get access to the knowledge graph. Furthermore, they can submit their code for their analyses to the system. The code will be executed on a High-Performance Cluster (HPC) and users can receive the results later on. In this case, Flowable can take advantage of PIDs for digital objects identification and management to facilitate the communication with the HPC system. As one may already notice, the whole process can be expressed as a workflow. A workflow, including error handling and notification, has been created and deployed. Workflow execution can be triggered manually or after predefined time intervals. According to our evaluation, the Flowable platform proves to be powerful and flexible. Further usage of the platform is already planned or implemented for many of our projects.

[†] Corresponding author: Triet Ho Anh Doan (Email: triet.doan@gwdg.de; ORCID: 0000-0002-7247-9108).

1. INTRODUCTION

Göttingen University Library, founded in 1734, is one of the five largest libraries in Germany [1]. It has a huge amount of text resources stored in different repositories, such as GDZ[①], Goescholar[②], and Textgrid[③]. Since the resources are scattered in many repositories, it is challenging for people to access them. In addition, the access to raw data is still managed manually. If users wants to have a raw text resource, they must first contact the librarian. Then, the librarian will contact the responsible person of the corresponding repository to get the requested data. After that, the requested data is sent back to the users. As one can see, the current scenario poses a difficult problem to the data gathering process. This difficulty makes it impossible to build a complete automatic text analysis workflow.

For that reason, it is necessary to have a simpler way for people to get access to the data. A service is developed which allows users to perform a wide range of actions, such as full-text search, resources download, and data analysis on High Performance Cluster. Behind the scene, this service collects data from various repositories and offer them to users. Since data are available in different format at different location, which can be accessed via different methods, this data gathering process is more complicated than it sounds. Following the high-cohesive and loose coupling design principle, each action, e.g. data downloading, standardising, indexing..., is treated as an independent module. Depending on specific use cases, appropriate workflows are developed by connecting those modules together. The first step in workflow creation is to choose its format. In our case, it is decided to use Business Process Model and Notation (BPMN), a highly standardised and well-supported workflow language.

The paper is structured as following: Section 1 explains the current scenario and the motivation of the project. Section 2 and 4 give insight about Flowable and the project architecture respectively. How Flowable could be used to support Canonical Workflow Framework for Research (CWFR) concepts is discussed in Section 5. Finally, the conclusion is presented in Section 6.

2. WHAT IS FLOWABLE?

To avoid misunderstanding since there are many products from Flowable, it is necessary to be clear that this paper is talking only about Flowable open source[④], which is a workflow management platform and a Java business process engine. Figure 1 gives an overview of the Flowable architecture. It encompasses four main components: Modeler, Task, Identity Management (IDM), and Admin. Each of them has its own role in the system and can be accessed under different user rights, which is configured via the IDM component.

① https://gdz.sub.uni-goettingen.de/
② https://goedoc.uni-goettingen.de/
③ https://textgrid.de/en/
④ https://flowable.com/open-source/

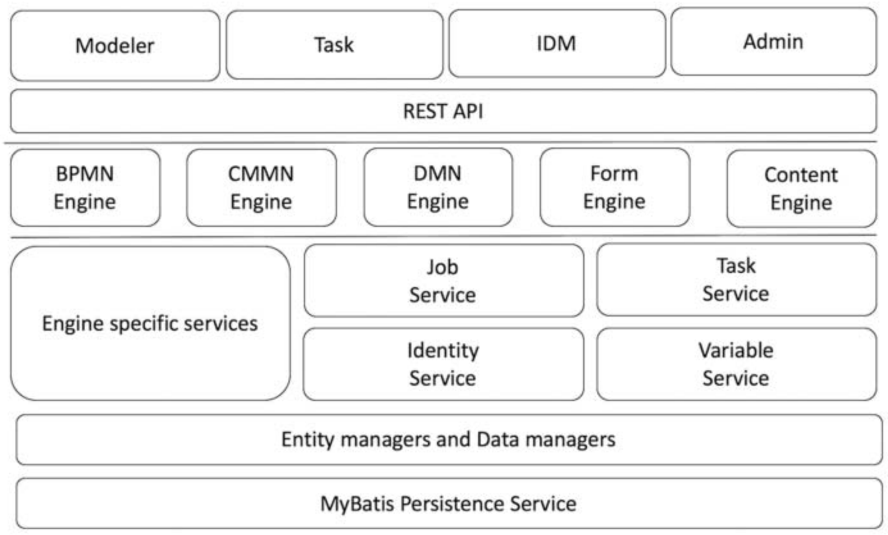

Figure 1. The architecture of Flowable [2].

All components expose various REST API endpoints for both public and cross-component communication. However, using REST API is not the only option to interact with Flowable because it is also shipped with an intuitive web user interface (UI). Via the UI, users can fully exploit all features offered by Flowable. As a workflow developer, one can work with the Modeler component while the users, who want to execute the workflows, can just focus on the Task component. A workflow can be easily created with drag-and-drop actions supported by the web UI. Behind the scene, all workflows are serialised into XML with BPMN schema. The IDM component is responsible for identity management. It can either use a local database or connect to a Keycloak[⑤] instance to enable Single Sign-On (SSO) feature. Last but not least, the system configuration is done in the Admin component.

As depicted in Figure 1, there are five engines running in Flowable: BPMN, CMMN, DMN, Form, and Content engine. For that reason, Flowable supports not only BPMN, but also Case Management Model and Notation (CMMN) and Decision Model and Notation (DMN). Figure 2 shows an example BPMN workflow developed using Modeler component of Flowable. The workflow contains two User tasks, one Script task, and one Decision task. Thanks to the Form and Content engine, workflow developers can create forms and link them to User tasks to get users' input. The Script task executes custom script in the chosen programming language and the Decision task can make decisions based on a predefined DMN model.

⑤ https://www.keycloak.org/

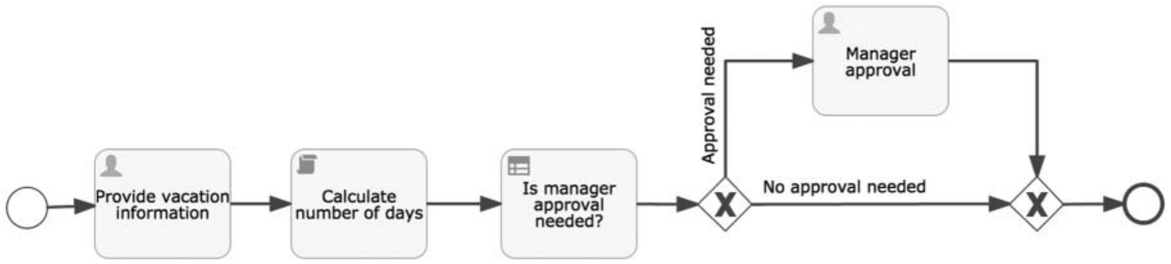

Figure 2. An example BPMN workflow developed with Flowable [3].

3. OTHER WORKFLOW MANAGEMENT SYSTEMS

Before deciding to use Flowable, other workflow management systems were also considered. The most prominent candidates among them are Apache Airflow, Prefect, Camunda, and Activiti.

Apache Airflow and Prefect are two powerful workflow management systems written in Python and quite similar to each other in many aspects. To create a workflow, users have to write some Python code to define tasks and how they are connected to each others. The whole workflow must form a Directed Acyclic Graph (DAG). Although these systems are really well-developed, they do not fit to our use cases. The fact that workflows are created by writing code is challenging, since our workflow creators are more domain knowledge experts instead of programmers. Using DAGs to represent workflows is not appropriate for complicated workflows because loops are not allowed in DAG. Besides, unlike BPMN, DAG is not a standardised workflow language. Finally, tasks which require human interaction are not well-supported by these systems.

On the other hand, Flowable, Activiti, and Camunda share some similarities because they come from the same code base. There was only Activiti at the beginning, then in 2013, Camunda split from the Activiti project [4]. Some years later, another fork was made from Activiti and we have Flowable. Instead of DAG, these systems represent workflows using BPMN. Furthermore, users can workflows simply by drag and drop on their web browsers. Although all these three systems meet our need, Flowable was chosen because it is more developed than Activiti [5] and it has many more stars on GitHub than Camunda.

4. INFRASTRUCTURE FOR TEXT DATA MANAGEMENT

4.1 System Architecture

As described in Section 1, the fact that data are stored at different repositories makes it extremely difficult for users to access them. Therefore, we develop a system to facilitate that process. Figure 3 illustrates the architecture of our system. There are three main components in the architecture, which are explained in this paper: *connector*, *knowledge extractor*, and *web server*.

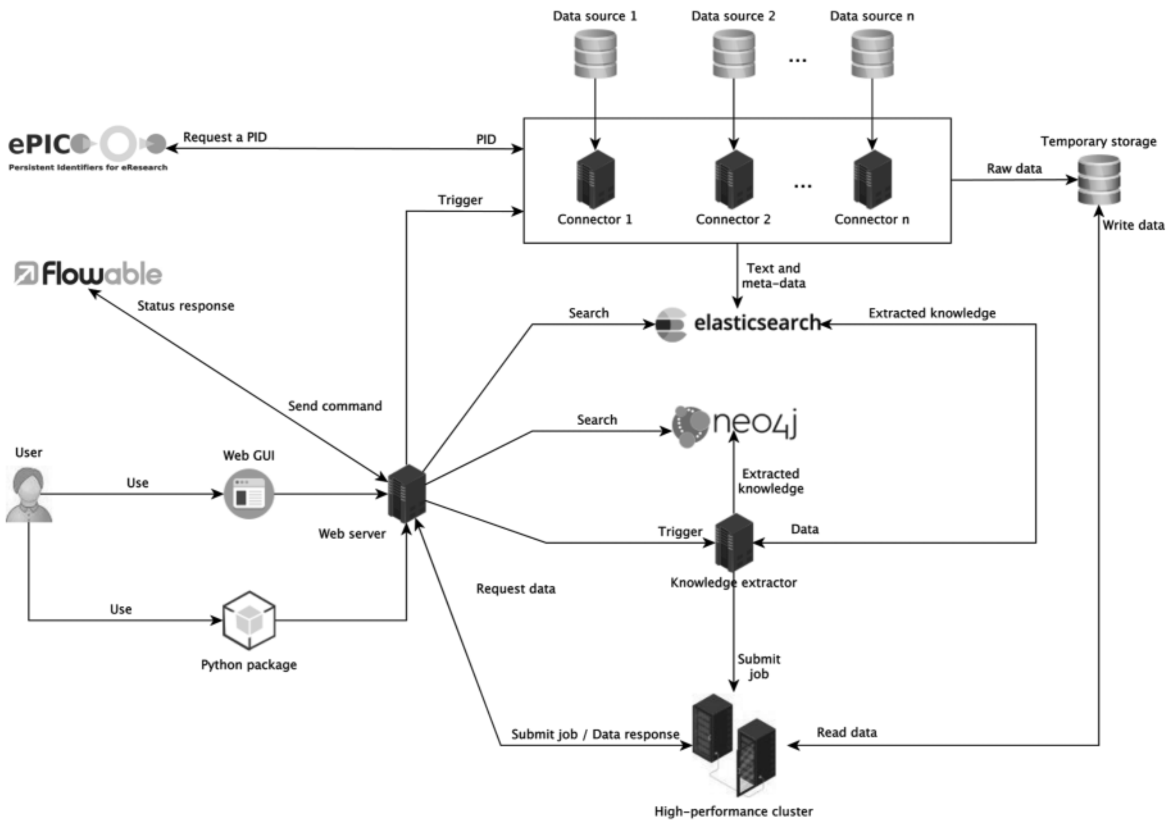

Figure 3. The system architecture, which shows all relevant components and how they interact with each other.

4.1.1 Connector

Connectors are the bridges between the system and the data sources. With each data source, a corresponding connector is developed. A connector is responsible for all actions related to its connected data source. These actions include but not limit to authentication against the data source, standardise metadata, requesting Persistent Identifiers (PID), downloading raw data, indexing data into Elasticsearch, keep the system up-to-date with changes from the data source, and so on. All metadata are stored as they are in Elasticsearch. Additionally, a set of metadata including title, author, abstract, and so on, is defined. The metadata in this set are extracted from the data sources and indexed into Elasticsearch with proper data type to allow searching.

According to the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) standard, each item in the repository has a unique identifier [6]. However, not all items from all data sources have identifiers, and even if they do, the identifier formats are different between data sources. To make sure that all items in the system can be uniquely identified in a consistent way, the connectors get items from data sources and assign PIDs to them. The PID is generated and managed by the European Persistent Identifier Consortium (ePIC)[⑥].

[⑥] https://www.pidconsortium.net/

4.1.2 Knowledge Extractor

The data obtained by connectors are stored in a central data storage. In this case, Elasticsearch, a feature-rich, powerful, and reliable platform with a strong focus on the search feature, is chosen. After the data is indexed into Elasticsearch by connectors, the knowledge extractor can read the data from Elasticsearch and run further analyses to gain insight into the data. One of the analyses is named-entity recognition. The entities are then linked together to build a graph, which is stored in the Neo4j graph database. Other analyses currently include topic modeling, sentiment analysis, and word cloud generation. However, this list will be extended as the time goes on. Depending on the size of the data, the knowledge extractor can decide if it will run the analyses locally or submit the job to the HPC. In the end, the outputs of the analyses are used to enrich the graph database and the metadata stored in Elasticsearch to improve the search function.

4.1.3 Web Server

Web server is the only entry point where the clients can interact with the system. As a user, one can use the system via a web UI or a Python package, which is developed together with the system. Searching for text resources, getting the raw data, viewing the entity graph, and submitting analyses, which will be executed on the HPC, are features supported by the web server. As illustrated on Figure 3, the web server can also trigger the connectors and the knowledge extractor. Although the system administrators can trigger these actions by sending commands directly to the web server via the REST API, these commands often come from Flowable. Additionally, when users want to run analyses on the text stored in the system, they can send requests to the web server, which will then be transformed and forwarded to the HPC. Users can check the status of their jobs any time via the web UI. For the HPC, when it needs to get the text, it has to send requests to the web server since the HPC locates outside of the system, which means that it is not allowed to access the Elasticsearch directly.

4.2 Role of Flowable in the System

From what described above, one can easily realise that a lot of tasks in the system can be represented as processes, or workflows, which contains various actions linking with each other. For example, to import a new data source into the system, it has to go through some certain steps, such as downloading data, extract metadata, and pushing data to Elasticsearch. Although these actions can be triggered by the web server independently, the responsibility of linking them together falls on Flowable. Therefore, Flowable can be considered as an orchestrator of the system.

It is necessary to mention that Flowable is a multi-tenant application. For that reason, the one presented in Figure 3 is not a dedicated instance for the system, but a Software as a Service which is shared by other systems as well. There are some workflows developed in Flowable and used in the presented system. Figure 4 shows one of them. Although this workflow was designed and deployed using Flowable version 6.5.0, it should work seamlessly with the latest version of Flowable at the moment, which is 6.7.1

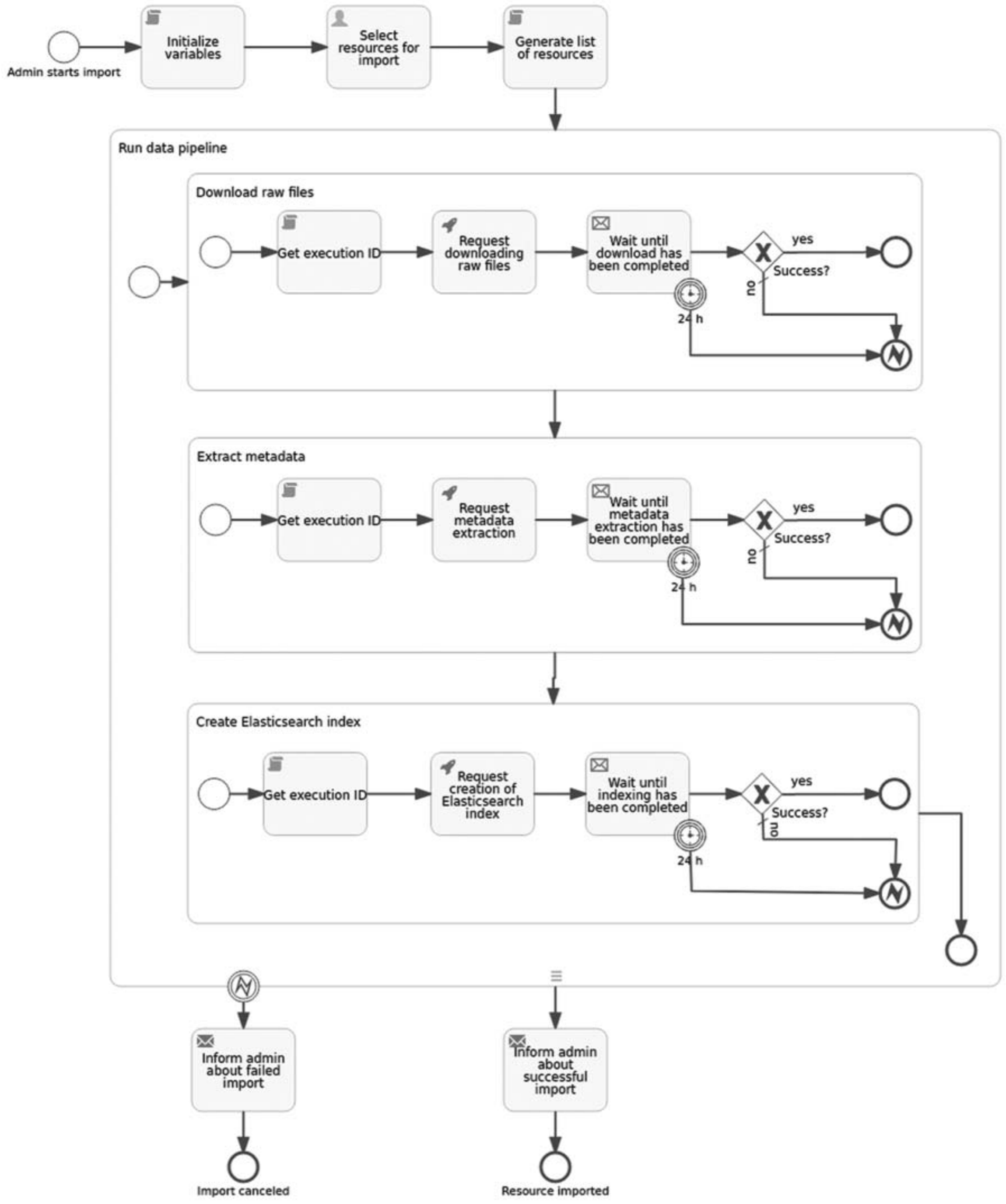
Admin starts import
Initialize variables
Select resources for import
Generate list of resources
Run data pipeline
Download raw files
Get execution ID
Request downloading raw files
Wait until download has been completed
24 h
yes
no
Success?
Extract metadata
Get execution ID
Request metadata extraction
Wait until metadata extraction has been completed
24 h
yes
no
Success?
Create Elasticsearch index
Get execution ID
Request creation of Elasticsearch index
Wait until indexing has been completed
24 h
yes
no
Success?
Inform admin about failed import
Import canceled
Inform admin about successful import
Resource imported

Figure 4. This workflow is developed with Flowable. It is triggered when the system administrator decides to import a new resource or update an imported resource.

At the beginning of the workflow, the administrator can decide which resource should be imported or updated, then the pipeline starts, which includes three sub-processes: file downloading, metadata extraction, and data indexing. For each sub-process, Flowable send a request to the web server, then the web server triggers the corresponding component to perform the task. On the Flowable side, after sending the request, the workflow stops and wait for further information from the web server. When the task is finished, the web server notifies Flowable the status of the task, if it is successful or failed. As depicted in Figure 4, the next sub-process starts only when the previous sub-process finishes successfully. Additionally, if the web server does not send any information after 24 hours, the sub-process is automatically considered as failed. In any cases, the administrator will receive an email regarding the final status of the workflow.

Since the data sources are changed with different frequency, i.e. some are more frequent than the others, the automatic versions of this workflow are executed at different frequency, depending on to which resources they are referring.

5. FLOWABLE AND THE CANONICAL WORKFLOW FRAMEWORK FOR RESEARCH

The CWFR is a concept to overcome the current limitations for processing pipelines currently in use. As described in [7], the CWFR adds another layer on top of the workflow execution machinery and framework, and the technical workflow framework. The basic elements of CWFR are described as recurring patterns, library of canonical steps, and libraries of domain specific specialised packages per step. To integrate all these elements into a larger framework a connection, "the glue", as it is called, needs to be added in between. Here the concept of FAIR Digital Objects (FAIR-DO or CWFR-DO) comes into play. If the output of each step is assigned with a PID and enhanced with the mandatory metadata, then the next processing step can read this metadata and use the digital object accordingly. The connection between the processing steps is therefore the appropriate use of PIDs and the required metadata (e.g., PID Kernel Information [8]).

The concept of a CWFR can be implemented using the open-source software Flowable. As described in Section 2, Flowable is the technical runtime environment for workflows described using BPMN. In the picture of CWFR, it covers at least the workflow execution machinery and framework. With its interfaces, it is possible to include other arbitrary technical workflow frameworks or tools and functions implemented by the researchers. Flowable allows to automate processing steps, allows iterating and has the opportunity to easily integrate user interactions using forms. Processing steps can be described as scripts (e.g., in Java) or as API calls to user defined functions. These can either be recurrent canonical steps or dedicated domain-specific steps.

5.1 CWFR-DOs in Flowable

Therefore, minting PIDs for the creation of CWFR-DO can be achieved either as part of the user defined function or by a dedicated step in the Flowable BPMN workflow. For the latter, a sub-workflow can be designed that aggregates metadata, either via forms or automatically, and conducts an HTTP-API call to the PID system. This sub-workflow could then be called from any point of the data processing workflow.

Metadata collected can be handled within Flowable so that the user-interactions can be reduced at later processing stages. A scientific workflow could start by reading the PIDs of the initial digital objects, e.g. by user interaction or from file. The initial PIDs and the newly created PIDs pointing to CWFR-DOs can be kept in memory of the active workflow and thus be used at different steps of the workflow.

5.2 CWFR with Flowable in Practice

Flowable, as technical base for CWFR, can be provided by a central IT service provider. Thus, maintenance of the runtime environment is not the duty of the scientific staff. The design and implementation of canonical steps, including the minting of CWFR-DO PIDs, should be conducted by an IT expert working for the service provider. The libraries of specialised packages can be built and maintained within collaborations between the service provider and the scientific communities. As design of workflows in Flowable is straightforward and good documentation exists, also researchers can be assigned to create new workflows.

6. CONCLUSION

This paper presents a system which allows users to easily access text data from multiple different repositories and run their own analyses on an HPC. Since the system architecture is designed using microservice approach, it consists of various components and 3rd-party services, and Flowable is one of them. Flowable plays the role of an orchestrator in the system, which coordinates all the independent actions via workflows to achieve certain goals, e.g. import new data source to the system. Finally, together with PIDs, Flowable can surely be used to support the concept of CWFR.

AUTHOR CONTRIBUTIONS

Triet Ho Anh Doan (doanhoanhtriet@gmail.com): Conceptualization, Methodology, Software, Writing—Original draft.

Sven Bingert (sven.bingert@gwdg.de): Conceptualization, Project administration, Writing—Review & editing.

Ramin Yahyapour (ramin.yahyapour@gwdg.de): Supervision, Funding acquisition.

REFERENCES

[1] Goettingen State and University Library. Available at: https://www.uni-goettingen.de/en/about+the+goettingen+state+and+university+library/56613.html. Accessed 13 July 2021

[2] Architecture—Flowable open source documentation. Available at: https://flowable.com/open-source/docs/cmmn/ch07-architecture/. Accessed 27 July 2021

[3] Flowable applications—Flowable open source documentation. Available at: https://flowable.com/open-source/docs/bpmn/ch14-Applications/. Accessed 28 July 2021

[4] Meyer, D., Simmons, D., Weidner, T., C. O. Team: Camunda engine evolution since Activiti Fork. Available at: https://camunda.com/blog/2016/10/camunda-engine-since-activiti-fork/. Accessed 24 November 2021

[5] Top 10 advances Flowable made since Activiti. Available at: https://www.flowable.com/blog/2021/02/top-10-advances-flowable-made-since-activiti/. Accessed 24 November 2021

[6] Lagoze, C., et al.: Open archives initiative—protocol for metadata harvesting—v.2.0. Available at: https://www.openarchives.org/OAI/openarchivesprotocol.html. Accessed 2 August 2021

[7] Hardisty, A., Wittenburg, P.: Canonical Workflow Framework for Research (CWFR)—position paper—version 2, December 2020. Working paper. Available at: https://osf.io/9e3vc/.

[8] Weigel, T., et al.: RDA recommendation on PID kernel information. Available at: https://www.rd-alliance.org/group/pid-kernel-information-wg/outcomes/recommendation-pid-kernel-information. Accessed 2 August 2021

AUTHOR BIOGRAPHY

Triet Ho Anh Doan obtained his Master's degree in 2018 in Applied Computer Science at the University of Göttingen, Germany. Since then, he has started working for the Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen as a researcher, software architect, and software developer. Many systems were designed and developed by him, such as a long-term preservation system for digitised library collections or a search engine for persistent identifiers. At the moment, he is pursuing his Ph.D. at the same university with the topic Big Data Infrastructure for Analysing Digitalised Library Collections. His research interests include research data management, persistent identifier, system architecture, and scientific workflow development.

Dr. **Sven Bingert** completed his Diploma in 2003 in Physics at the Karlsruhe Institute for Technology, Germany, and obtained his Ph.D. degree in 2009 in Astrophysics from the University of Freiburg, Germany, as a member of the Kiepenheuer Institute for Solar Physics. After four years as a Postdoc in the Coronal-Dynamics Group at the Max Planck Institute for Solar System Research, Dr. Bingert joined the Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen, Germany. Since 2021, he has been the Deputy Head of the eScience working group. His main research topics are research data management, persistent identifiers, and development of scientific workflow services.

Prof. Dr. **Ramin Yahyapour** holds a doctoral degree in Electrical Engineering. His research focus is on efficient resource management and its application on service-oriented infrastructures, clouds, and data management. Since October 2011, he has been the Managing Director of Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen, Germany. At the same time, he became full professor for Practical Computer Science at the Georg-August-Universität Göttingen. He is active in several national and international research projects, e.g., a scientific coordinator of the FP7 integrated project SLA@SOI and an executive member of the CoreGRID Network of Excellence. He is organizer and program committee member of several conferences and workshops as well as reviewer for multiple journals.